Task 2.1 – CounterTask

Program.cs

```
using System;

namespace CounterTask;

public class Program
{
    // PrintCounters method
    private static void PrintCounters(Counter[] counters)
    {
        foreach (Counter c in counters)
        {
            Console.WriteLine($"{c.Name} is {c.Ticks}");
        }
        Console.WriteLine();
    }

    static void Main(string[] args)
    {
        // Create an array of 3 counters
        Counter[] myCounters = new Counter[3];

        // Initialize counters with names
        myCounters[0] = new Counter("Counter 1");
        myCounters[1] = new Counter("Counter 2");
        myCounters[2] = myCounters[0];

        // Print initial state
        Console.WriteLine("Initial state:");
        PrintCounters(myCounters);
        // Increment counters different numbers of times
        for (int i = 1; i <= 9; i++)
            myCounters[0].Increment();
        for (int i = 1; i <= 14; i++)
            myCounters[1].Increment();

        // Print after increments
        Console.WriteLine("After increments:");
        PrintCounters(myCounters);

        // Reset the second counter
        myCounters[2].Reset();

        // Print after reset
        Console.WriteLine("After reset:");
        PrintCounters(myCounters);

        //
        Console.WriteLine("Testing IncrementByFive:");
        myCounters[0].IncrementByFive();
        myCounters[1].IncrementByFive();
        myCounters[2].IncrementByFive();
        PrintCounters(myCounters);
        //will result Counter 1 is 10 because myCounters[2] = myCounters[0]

        Console.WriteLine("Press any key to exit...");
        Console.ReadKey();
    }
}
```

Task 2.1 – CounterTask

Counter.cs

```csharp
using System;

namespace CounterTask;

public class Counter
{
    //Add private fields
    private int _count;
    private string _name;

    //Create constructor to initialize
    public Counter(string name)
    {
        _name = name;
        _count = 0;
    }

    //Add Increment method
    public void Increment()
    {
        _count++;
    }

    //Add Reset method
    public void Reset()
    {
        _count = 0;
    }
```

```
//Add Name property (read-write)
public string Name
{
    get
    {
        return _name;
    }
    set
    {
        _name = value;
    }
}

//Add Ticks property (read-only)
public int Ticks
{
    get { return _count; }
}

//Q12:Add ResetByDefault method
//Use unchecked block because value is too big for int
//unchecked prevents overflow exception
//The large value wraps around to a negative number due to overflow
public void ResetByDefault()
{
    unchecked
    {
        _count = (int)214748364881; //Given value with my student ID last 4 digits //4881
    }
}

//Q13: Method to increase count by 5
public void IncrementByFive()
```

```
    {
        _count += 5;

    //Q13 Answer: YES, the code still runs without bugs/crash
    //Reason: Adding 5 is a simple arithmetic operation that won't cause problems even with overflow
values
    }
}
```

Task 2.2 – ShapeDrawing

Program.cs

```
using System;

namespace ShapeDrawing;

public class Program
{
    public static void Main(string[] args)
    {
        //Declare a shape object
        Shape myShape;

        //Create a new shape object
        myShape = new Shape(181);

        //Draw the shape
        myShape.Draw();

        //Check if the shape is at the position (10,10)
        Console.WriteLine($"Is the shape at (10,10)? {myShape.IsAt(10,10)}");

    }
}
```

Task 2.2 - ShapeDrawing

Shape.cs

```
using System;

namespace ShapeDrawing;

public class Shape
{
    //Fields
    private string _color;
    private float _x;
    private float _y;
    private int _width;
    private int _height;

    //Create constructor
    public Shape(int param)
    {
        _color = "Color.Chocolate"; // As my name is Min Thu Kyaw Khaung, the first letter 'M' which is after
A-L.
        _x = 0.0f;
        _y = 0.0f;
        _width = (param);
        _height = param;
    }

    //Draw the shape
    public void Draw()
    {
        Console.WriteLine("Color is " + _color);
        Console.WriteLine("Position X is " + _x);
        Console.WriteLine("Position Y is " + _y);
        Console.WriteLine($"Position is ({_x},{_y})");
        Console.WriteLine("Width is " + _width);
        Console.WriteLine("Height is " + _height);
    }


    //Check if the shape is at the position (xInput,yInput)
    //IsAt method
    public bool IsAt(int xInput, int yInput)
    {
        return (xInput > _x && xInput < (_x + _width) && yInput > _y && yInput < (_y + _height));
    }

    //Properties
    public string Color
    {
        get { return _color; }
        set { _color = value; }
    }

    public float X
    {
        get { return _x; }
        set { _x = value; }
    }
    public float Y
    {
        get { return _y; }
        set { _y = value; }
    }
    public int Width
    {
        get { return _width; }
        set { _width = value; }
    }
    public int Height
    {
        get { return _height; }
        set { _height = value; }
    }
}
```